JavaScript replace: manipulando Strings y regex

Mario Souto

30/06/2021

En **JavaScript** el `String replace` se usa a menudo para hacer desde simples **sustituciones en Strings** a cambios complejos con **expresiones regulares**. Su uso es tan simple como `variable.replace('valor', 'sustitucion')`. Estoy seguro de que alguna vez has necesitado tomar un tramo de texto al final de un HTML o eliminar una palabra específica de un código.

En los formularios web es común tener que lidiar con el **campo NIT**. Lo que realmente importa cuando ponemos un campo como este son los números, pero al construir interfaces para el usuario final es muy importante mostrar al usuario el formato que está acostumbrado a ver en los documentos. Veamos cómo podríamos solucionar este problema con los poderes de **JavaScript** y tu biblioteca estándar.

Reemplazando palabras con String replace

Si queremos tomar los números brutos del usuario y convertirlos a la versión con el formato o mascara de campo. Una solución ultra directa para solucionar el problema, puede ser tomar el valor que queremos modificar y aplicar la función `.replace` de las strings encima de él.

Si estamos interesados en manipular un NIT, por ejemplo, podemos tomar el valor del NIT puro y decir que queremos convertirlo a su versión amigable para el usuario:

```
const NITSinFormato = 'nit es 25684677037'
 const NITFormateado = NITSinFormato.replace('25684677037', '256.846.770-37')
 console.log(NITFormateado)  *// El retorno será 'nit es 256.846.770-37'*
```

La función `.replace` recibe como parámetros el estándar que buscamos y como segundo parámetro lo que queremos colocar en su lugar.

En este caso, estamos prediciendo el futuro y creando un código asumiendo que el usuario tiene un cierto valor de NIT. Para este uso, el replace puede sonar muy raro, pero si nuestro objetivo es restringir alguna palabra, como un garabato, el uso anterior encaja como un guante:

```
const frase = 'Frase con una palabra-fea'
 frase.replace('palabra-fea', '********')
 console.log(frase) *// El retorno seria 'Frase con una ****')
```

Si desea ver estos ejemplos dentro de VSCode de una manera muy interesante, le recomiendo este plugin: https://quokkajs.com/

Reemplazando todas las ocurrencias con el apoyo de Regex

La idea anterior de eliminar palabras de un texto funciona muy bien, pero si la palabra aparece dos veces en el texto, empezaremos a tener problemas:

```
 const frase = 'Frase con una palabra-fea y hay otra palabra-fea al final'
```

```
const fraseActualizada = frase.replace('palabra-fea', '********')  console.lo
```

Observe que en la segunda ocurrencia de la palabra que queremos eliminar de la frase, la función replace no hizo nada. Esto se debe a que cuando el primer parámetro de la función de replace es una string, siempre busca la primera ocurrencia en el texto base para el valor buscado. Para tener algo más dinámico necesitaremos recurrir al poder de las Expresiones Regulares, que en definitiva nos ayudan a encontrar estándares en el texto de una forma mucho más completa.

Para solucionar nuestro problema anterior, simplemente cambia el primer parámetro pasado al relace de `'palabra-fea'` a `/palabra-fea/g`. La `g` al final de la regex indica que queremos buscar globalmente en el texto base.

```
const frase = 'Frase que empieza con una palabra-fea y hay otra palabra-fea a
const fraseActualizada = frase.replace(/palabra-fea/, '********')
console.log(fraseActualizada) *// "Frase que empieza con una **** y hay otra
```

Regex: expresiones regulares

Muchos desarrolladores, cuando necesitan trabajar con Regex (*expresiones regulares*), suelen pensar "*Siempre aprendo, pero cuándo lo voy a usar me olvido enseguida*". En el día a día, en la mayoría de los casos, no usamos Regex y si lo vamos a usar, podría valer la pena una lib de validación como Yup o Joi. En cualquier caso, recordemos que será fundamental poder tratar el caso del NIT aquí en la publicación:

Regex buscan patrones, la forma más directa es escribir tu propia palabra que quieras encontrar:

/palabra/

lo mismo funciona para los números:

/ 12345679810 /

Sin embargo, si desea algo más amplio con caracteres especiales y todo, donde el orden no importa, deberás definir un range de caracteres:

A-Z0-9! -\

Y en el ejemplo anterior, para obtener más de un carácter, es necesario declarar cuántas ocurrencias estás buscando:

[A-Z0-9! -] {3}

También es posible simplificar para capturar solo caracteres alfanuméricos puedes usar:

| w

Ya para capturar caracteres relacionados con dígitos, puedes utilizar:

| D

Y si deseas capturar grupos dentro de un match de regex para poder trabajar mejor como veremos en los siguientes ejemplos, podemos usar los paréntesis:

(| d {4}) - (| d {4})

¿Hay otros casos como el uso de "?" para que alguna parte de la regex sea opcional, los marcadores de comienzo "^" y fin "$", pero todo eso produciría una serie de publicaciones o incluso un curso.

Si quieres profundizarte, te dejo esta guía de bolsillo que siempre uso y si realmente quieres aprender a trabajar con expresiones regulares, te dejo esto como un consejo este curso de regex.

Rescatando group matches con la función replace

Ahora que recordamos un poco más sobre las expresiones regulares, podemos usar la idea de buscar estándares encajando en grupos y usando el segundo parámetro de `.replace`, el "$"seguido del número correspondiente al grupo en el orden en que se escribió la Regex:

```
const nit = '12345679810'
 const nitFormateado = nit.replace(/(\d{3})?(\d{3})?(\d{3})?(\d{2})/, "$1.$2.$
 console.log(nitFormateado) *// El retorno sería 256.846.770-37*
```

Si desea hacer algo más complejo y simplemente pasar una string rescatando los grupos no sea suficiente, también puedes pasar una función en el lugar de la string en el segundo parámetro:

```
const nit = '12345679810'
 const nitFormateado = nit.replace(/(\d{3})?(\d{3})?(\d{3})?(\d{2})/, function
 console.log(nitFormateado) // El retorno sería 256.846.770-37
```

Dudas comunes en el manejo de strings con Regex

- Capturando contenido entre apertura y cierre de una tag. Ten cuidado en esos casos. HTML es una estructura que es imposible de analizar perfectamente a través de regex, elige siempre usar la API DOM para manipular HTML o una biblioteca como JSDOM para convertir un texto a la estructura de árbol y facilitar la extracción de contenido.
- Dando match en slugs
- Lidiando con hexadecimales
- Trabajando con URL

Métodos de String

Hay muchos métodos esenciales en la String, algunos otros interesantes:

- split: Es muy útil cuando deseas dividir una string en una o más partes agrupadas en una array.
- trim: Muy usada al recibir un input del usuario para eliminar los espacios en blanco antes del comienzo de la string y después del último carácter que no sea un espacio en blanco.
- includes: Esta es mi función favorita de las strings, devuelve true o falsw si hay o no, respectivamente, en la string base el valor que das como argumento al .includes().

Bibliotecas de manipulación de Strings

- Existe una lib muy popular para manejar fechas llamada MommentJS https://momentjs.com/, es necesario tener algunos cuidados a la hora de usarla, así que dejo el dato de este otro enlace que quizás tenga alguna alternativa que te pueda servir tan bien como la otra https://github.com/you-dont-need/You-Dont-Need-Momentjs;

- Si quieres lidiar con máscaras y formateo (ayuda con nit, teléfono, nit pj ...), dejo esta lib https://www.npmjs.com/package/mask-js;

- Libs súper geniales para manejar la validación de schema: Yup o Joi;

- No es el enfoque de la publicación, pero lidiar con el formato y las operaciones sobre dinero puede ser muy complicado. https://sarahdayan.github.io/dinero.js/;

Si deseas asegurarte de que un input del usuario sea realmente seguro para guardarlo en tu base de datos, es importante usar una lib que haga la función de eliminar posibles contenidos que generarían inyección de scripts como ésta https://github.com/apostrophecms/sanitize-htmlSi tienes curiosidad como yo, te dejo el dato para ver este sitio genial https://regex101.com que estoy seguro te ayudará mucho a crear regex de una manera rápida y fácil de probar.

En esta publicación abordaremos una serie de temas importantes relacionados con la manipulación de strings. Vimos los poderes de `replace`, casos comunes de tratamiento de strings y espero que esta publicación ayude tanto a ti como a algún amigo tuyo que pueda estar sufriendo ahora queriendo **validar un NIT** o cualquiera de los ejemplos que vimos anteriormente.

Si te gustó este artículo te invito a conocer los cursos de Front-End y JavaScript en la plataforma de Alura Latam.

Puedes leer también:

- Convirtiendo String a número en JavaScript
- Cambiando CSS con JavaScript
- ¿Cómo funciona el import y export de JavaScript?

ARTÍCULOS DE TECNOLOGÍA > FRONT END

¡QUIERO EMPEZAR A ESTUDIAR!

Paga en moneda local en los siguientes países

ANUAL

US$79,90

un solo pago de US$79,90

- 218 cursos
- Videos y actividades 100% en Español
- Certificado de participación
- Estudia las 24 horas, los 7 días de la semana
- Foro y comunidad exclusiva para resolver tus dudas
- Acceso a todo el contenido de la plataforma por 12 meses

¡QUIERO EMPEZAR A ESTUDIAR!

Lógica de Programación | Java

Cursos de Front End

HTML y CSS | JavaScript | React

Cursos de Data Science

Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps

Docker | Linux

Cursos de Innovación y Gestión

Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics | Liderazgo y Gestión de Equipos | Startups y Emprendimiento